基于功能匹配分解的协议转换器构造技术研究*)

赵新有　古天龙　蔡国永　董荣胜
(桂林电子工业学院计算机系　桂林541004)

Research of Protocol Converter Based on Function Mapping Partition

ZHAO Xin-You　GU Tian-Long　CAI Guo-Yong　DONG Rong-Sheng
(Department of Computer, Guilin Institute of Electronic Technology, Guilin 541004)

Abstract　A protocol converter can solve the problem of communication between heterogeneous networks. Based on function matching partition, this paper proposes a novel approach to construct the converter, which consists of two processes and can translate messages concurrently. This technique can improve the conversion efficiency much. The illustrative example of AB and SW protocols is also presented.

Keywords　Communicating finite state machines, Protocol conversion, Heterogeneous networks

1　引言

异构网络互联的一个技术难点就是协议的不匹配，它导致不同网络体系下的用户难以直接相互通信。为了解决此困难，在协议间插入转换器当作通信中介，它从一个协议接收信息，经转换后发送给另一个协议，实现不同网络之间的通信。

在协议转换模型中，分为单进程与双进程转换模型。在单进程转换模型中，利用单个转换进程充当转换器，实现异构协议间的通信。在转换过程中，转换器要与协议的通信进程相互通信，每个进程都要受到其它进程的限制，因此实现转换器并行转移比较困难[1~3]。在双进程转换模型中，利用两个转换进程充当转换器的角色，实现报文的转移。Shu 等[4]通过在协议通信进程中插入 PUT-GET 操作，形成四个进程的异构系统，可以使进程间并行运行。后来，Shu 等[5,6]又进一步改进了此方法，但在通信进程中插入 PUT-GET 操作时仅描述了单个转移，方法仅可以实现一对报文间的并行转移[1]，不能充分实现协议间的并行转移。

基于此，本文对双进程转换模型做了进一步的研究，给出了一种双进程转换方法。此方法构造的转换器可以提高异构协议系统的并行性，使转换器的转换效率得到提高。

2　协议与功能匹配描述

协议的形式描述是协议设计的一个关键阶段。在目前已开发的形式描述技术(FDT)中，通信有限状态机(CFSM)由于具有简单性、直观性和高度抽象性，且易于自动实现，被广泛地应用于协议的形式描述、验证中[1,3,7]。

定义1　通信有限状态机(CFSM)A 是一个四元组，$A=(S_A, \sum_A, s_{0A}, \delta_A)$，其中 S_A 是状态集；$\sum_A$ 是报文集，包含发送报文与接收报文，分别用 $-e(+e)$ 来表示(在此+/−仅仅为了分析的方便，不会增加报文的数量)；$s_{0A}\in S_A$，是 A 的初始状态；δ_A 是部分转移函数，即 $\delta_A: S_A\times\sum_A\rightarrow S_A$ 的一个子集，每个转移记为 $\delta_A(s,m)$，它表示 $CFSM$ 在状态 $s\in S_A$，执行报文 $m\in\sum_A$ 所达的状态。

定义2　协议 P 是一个二元组，$P=[P_0,P_1]$，其中每个 $P_i(i=0,1)$ 为协议的一个通信进程，分别为协议的发送进程与接收进程，采用 CFSM($P_i=(S_{P_i}, \sum_{P_i}, s_{0P_i}, \delta_{P_i})$)来描述。

对于不同的两个协议 $P=[P_0,P_1]$ 与 $Q=[Q_0,Q_1]$，为使通信进程 P_0 与 Q_1 相互通信，需要选取 P_1 与 Q_0 构造转换器 C，用转换器 C 取代 P_1 与 Q_0，插入转换器 C 后形成新的协议系统 $D=(P_0,C,Q_1)$，此时指向初始通信进程 P_1 与 Q_0 的报文将直接指向转换器 C。

为描述待转换协议报文间的转移关系，在此采用功能匹配来描述。功能匹配描述协议间的相关报文(即两协议间必须进行转换的报文，如在数据传输服务中，两协议的数据传输部分)如何进行通信，使最终的转换器依此描述进行协议间的报文转移[4,5,9]。依据协议转换的需求，由协议设计者找出相关报文，然后构造出功能匹配：在本文中，功能匹配 M 采用 CFSM 进行描述，$M=(S_M, \sum_M, s_{0M}, \delta_M)$，描述协议中通信进程 P_1 与 Q_0 的相关报文序列的转移关系，它的报文集 $\sum_M\subseteq\sum_{P_1}\cup\sum_{Q_0}$。

3　转换器的构造过程

下面采用双进程转换模型来构造协议转换器。用 C_P 与 C_Q 表示模型中的两个转换进程，P_0 与 Q_1 通过它们可以相互通信。由于功能匹配 M 描述了通信进程 P_1 与 Q_0 之间的相关报文的转移关系，依相关报文的来源将功能匹配 M 分解为两部分 M_P 与 M_Q，分解时通过在 M_P 与 M_Q 中插入辅助转移保持 M 报文的转移序列；最后利用 M_P、M_Q 分别与 P_1、Q_0 复合得到转换进程 C_P、C_Q。在分解中加入辅助转移有两个作用：1)它们执行 C_P 与 C_Q 之间数据的转换；2)在 C_P 与 C_Q 之间处理同步操作。

3.1　功能匹配的分解

*)本课题得到国防预研基金项目及广西自然科学基金项目的资助。赵新有　研究生，研究方向为形式化技术、网络协议。古天龙　教授，博导，研究方向主要为形式化技术、符号模型检验、协议工程。蔡国永　副教授，研究方向主要为形式化技术、软件测试、验证。董荣胜　副教授，研究方向主要为形式化技术、分布式实时调度、计算机科学与技术方法论。

考虑 M 中状态 s 的一个转移报文 $e\in\sum_M\subseteq\sum_{P_1}\cup\sum_{Q_0}$，依 e 的来源(是属于 $\sum_{P_1}$ 还是属于 $\sum_{Q_0}$)把状态 s 分为两部分，对每一部分的报文来说，又分为输入与输出，所以又把每一部分分为两部分，这样状态 s 就被分成了四部分，每一部分用一个子状态来描述，这样状态 s 就被拆分为四个子状态 s_1,s_2,s_3,s_4(如图1所示)。

子状态 s_1 与 s_2(s_3 与 s_4)的转移报文来自于 $\sum_{P_1}(\sum_{Q_0})$，由于它们之间的转移次序将由 $M_P(M_Q)$ 保存，在此没必要在它们之间插入辅助转移，用空转移 λ 连接；又由于 s_2 与 s_4 的输出必须后于 s_3 与 s_1 的输入，因此执行状态 s_2 与 s_4 输出之前必须等待 s_3 与 s_1 的输入，因而需要在此加入辅助转移以使它们得到同步(如图1中从状态 s_1 到 s_4 的转移 $(P\rightarrow Q)x$ 和 s_3 到 s_2 之间的转移 $(Q\rightarrow P)y$。在此 $(A\rightarrow B)x$ 表示若 B 需与 A 同步，用报文 x 来实现)。依此分析，下面给出分解 M 的方法：

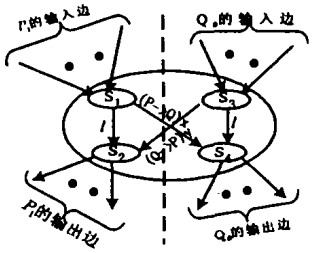

图1　状态 s 的分解示意图

Step 1　对 M 中的每个状态 s 执行下面的四步：

① 依状态 s 转移报文的来源(是属于 $\sum_{P_1}$ 还是属于 $\sum_{Q_0}$)与报文的转移方向(输入与输出)把 s 分为四个部分；

② 用子状态 $s_i(i=1,2,3,4)$ 替换每一部分，这些子状态用来插入辅助转移，其中 $s_1(s_3)$ 接收来自 $P_1(Q_0)$ 报文，$s_2(s_4)$ 发送来自 $P_1(Q_0)$ 报文；

③ 依据下面的规则连接四个子状态：

I 若状态 s_1 有输入及 s_2 有输出，则用 λ 连接 s_1 到 s_2；若状态 s_3 有输入及 s_4 有输出，则用 λ 连接 s_3 到 s_4；

II 若状态 s_1 有输入及 s_4 有输出，则在 s_1 到 s_4 之间创建辅助转移 $(P\rightarrow Q)x$；若状态 s_3 有输入及 s_2 有输出，则在 s_3 到 s_2 之间创建辅助转移 $(Q\rightarrow P)y$。

④ 在四个子状态中，若某子状态与其它子状态之间没有辅助转移，则应删除它。

Step 2　赋初始状态：

① 如果 s_{0M} 仅有 P_1 或 Q_0 的输出边，则此初始状态为新状态的初始状态；

② 如果 s_{0M} 既有 P_1 又有 Q_0 的输出边，则新创建的状态都是初始状态。

Step 3　构造 M_P 与 M_Q：

① 把所有与 Q_0 有关的转移置为 λ，变换所有辅助转移 $(P\rightarrow Q)x$ 置为 $-x$，所有辅助转移 $(Q\rightarrow P)y$ 置为 $+y$；删除所有标有 λ 的边；合并连接 λ 的两个状态为一个状态，把与这两个状态相连的所有边与合并状态相连，即可产生 M_P；

② 用相似的方法可以构造出 M_Q。

3.2　转换进程 C_P 与 C_Q 的构造

取得 M_P 与 M_Q 之后，共有六个 CFSM P_0,P_1,M_P,M_Q,Q_0,Q_1。P_0 与 P_1、M_P 与 M_Q、Q_0 与 Q_1 之间可以直接通信，但 M_P 与 M_Q 还不能充当转换进程，因为它们还不能使 P_0 与 Q_1 相互通信。

为达到通信目的，须让 M_P 与 M_Q 与初始协议的通信进程 P_1 与 Q_0 进行复合形成转换进程 C_P 与 C_Q。由于 M_P 中部分报文来自于 P_1，把二者之间的重复报文合并，不同的报文保留，可形成新的 $CFSMC_P$，C_P 不仅可以与 P_0 相互通信，也可以与 M_Q 通信。同理可以取得 C_Q。在转换过程中，当 $C_P(C_Q)$ 执行 M_P 或 M_Q 中的辅助转移时，C_P 与 C_Q 才需要等待对方的数据包(同步)，其它时间它们都可并行执行。给两个 CFSM $A=(S_A,\sum_A,s_{0A},\delta_A)$ 与 $B=(S_B,\sum_B,s_{0B},\delta_B)$，对二者进行复合将会产生一个新的 CFSM $Z=(S_Z,\sum_Z,s_{0Z},\delta_Z)$，下面给出产生 Z 的方法：

Step 1. 输入 CFSM A 与 B，初始化 $S_Z=Nil$，$\sum_Z=Nil$，$\delta_Z=Nil$；

Step 2. 进行 $S_A\times S_B$ 操作为状态集 S_Z，状态用 $[p,q]$ 表示，其中 $p\in S_A,q\in S_B$；

Step 3. 置状态 s_{0Z} 为 $[s_{0A},s_{0B}]$；

Step 4. 对 $\sum_A$ 与 $\sum_B$ 进行并操作，结果作为 Z 的报文集，即 $\sum_Z=\sum_A\cup\sum_B$；

Step 5. 在状态之间创建转移：

① 如果对 $\forall\ p\in S_A,m\in\sum_A,m\notin\sum_B$，有 $\delta_A(p,m)=p'$，则在 δ_Z 中加转移 $\delta_Z([p,q],m)=[p',q]$；

② 如果对 $\forall\ p\in S_B,m\in\sum_B,m\notin\sum_A$，有 $\delta_B(q,m)=q'$，则在 δ_Z 中加转移 $\delta_Z([p,q],m)=[p,q']$；

③ 如果对 $\forall\ p\in S_A,\forall\ q\in S_B\ m\in\sum_A\cap\sum_B$，有 $\delta_A(p,m)=p'$，$\delta_B(q,m)=q'$，则在 δ_Z 中加转移 $\delta_Z([p,q],m)=[p',q']$；

Step 6. 删除死锁状态与没有定义接收的转移[4,7,8]；

Step 7. 结束算法，输出 Z。

应用以上方法，即可取得转换进程 C_P(通过 P_1 与 M_P 复合)与 C_Q(通过 Q_0 与 M_Q 复合)，它们一起完成协议间报文的转换。

4　应用

为说明方法的有效性，以协议 AB(用 $P=[P_0,P_1]$ 表示)与 SW(用 $Q=[Q_0,Q_1]$ 表示)为例进行说明。AB 协议的发送端 P_0 每发送一个数据包 d_i，就等待响应报文，收到此包的响应报文 a_i 后，再发送下一个数据包；若超时没有收到报文 a_i，则重发数据包。在协议的接收端，收到包后首先检查此包的序列号，是否是重传包，若是则丢弃，并要求传送下一个数据包；若不是重传数据包则将此包保存，并响应收到此包。SW 协议是在收到此包响应报文之前发送端 Q_0 一直发送数据包，直至收到报文 a 后，才开始发送新的数据包；接收端 Q_1 在收到数据包后回应报文 a。它们的 CFSM 模型如图2所示。

在本例中利用 AB 协议的接收端 P_1 与 SW 协议的发送端 Q_0 构造转换器。由于两个协议均提供数据传输服务，需要进行转换的是数据部分，因此 $-D$、$+d_0$ 及 $+d_1$ 是相关报文。Q_0 在没有收到从 P_1 发送的数据包 d_i 之前，不能发送数据包 $-D$，它们的执行次序是不能改变的，而其它的非相关报文的转移次序是不固定的，即可以在收到数据包之前发送确认报

文也可以在收到数据包之后发送确认报文，所以在此不用考虑它们。由此可构造 P_1 与 Q_0 间的功能匹配 M，它的 CFSM 模型如图3所示。

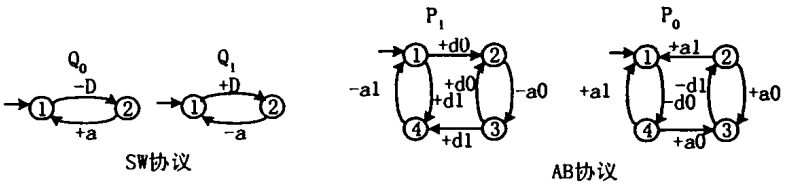

图2　协议的 CFSM 模型

对图3中的状态1，它的输入有 $+d_1$（P_1 的报文）与 $-D$（Q_0 的报文），输出有 $+d_1$ 与 $+d_0$（P_1 的报文），在此状态没有报文属于 Q_0 的输出，所以把状态1分解为三个状态[1,1]，[1,2]，[1,3]，在[1,3]到[1,2]之间加入辅助转移 $(Q\rightarrow P)t1$，在[1,1]到[1,2]用 λ 连接，在[1,2]到[1,1]用 $+d_1$ 连接。按同样的方法分解状态2,3,4。将分解后的 CFSM 中与 Q_0（P_1）相关的报文置为 λ，删除空转移 λ 后得到相应的 M_P（M_Q）（如图4所示）。应用复合算法将 M_P（M_Q）与 P_1（Q_0）进行复合得到转换进程 C_P（C_Q）（如图5所示）。

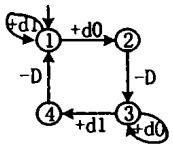

图3　功能匹配 M 的 CFSM 模型

为比较单进程与双进程转换模型转换器执行时间上的不同，在此给出一个周期内两模型的转换过程（单进程转换模型参考文[2,9]），比较结果如表1所示（在此假设每发送一个或接收一个报文将消耗一个时间单位，内部的转移不需要时间）。

单进程的转换器要转移这些数据一般需要8个单位的时间；Liu 的方法每转移一次，须利用 PUT 与 GET 进行同步，所以它的平均执行时间是6个单位时间。本文中，充分利用协议的自身属性，提高了转换器的转换效率，平均每传输一个数据包比一般的转换器可节省两个单位时间。

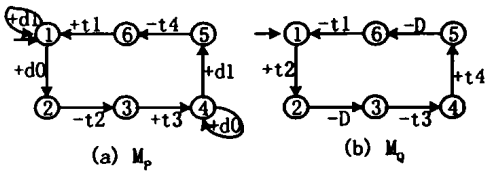

图4　M_P 与 M_Q 的 CFSM 模型

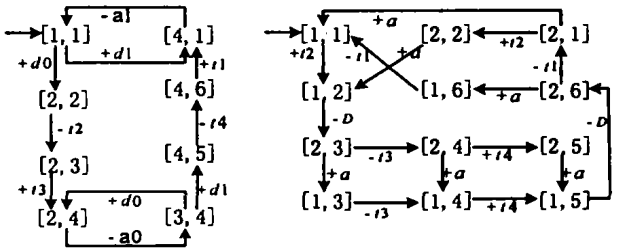

图5　最终取得的两个转换进程 C_P 与 C_Q

表1　转换过程时间对照表（E—通信进程，T—时间单位，C—转换进程，P(X)—表示 Liu 方法的 PUT(X)操作，G(X)—表示 Liu 方法的 GET(X)操作）

	双进程转换过程								单进程的转换过程		
	本文方法的转换过程				Liu 方法的转换过程						
E / T	P_0	C_P	C_Q	Q_1	P_0	C_P	C_Q	Q_1	P_0	C	Q_1
1	+a1/−d0		+a/−t3 +t4/−D		+a1/−d0		G(d1)/−D		+a1/−d0		
2		+d0/−t2 +t3/−a0		+D/−a		+d0/P(d0)		+D/−a		+d0/−D	
3	+a0/−d1		+a/−t1 +t2/−D				+a/P(a) G(d0)/−D				+D/−a
4		+d1/−t4 +t1/−a1		+D/−a		G(a0)/−a0		+D/−a		+a/−a0	
5	+a1/−d0				+a0/−d1		+a/P(a)		+a0/−d1		
6						+d1/p(d1) G(a1)/−a1				+d1/−D	
7					+a1/−d0		G(d1)/−D				+D/−a
8						+d0/P(d0)		+D/−a		+a/−a1	
9									+a1/−d0		

结束语　本文采用双进程转换模型取得异构协议的转

（下转第130页）

使用了基于 HASH 表存储的保护套件封装机制[9]。该机制可以解决目前多数系统中存在的以下问题：(1)由于两阶段协商涉及较多的状态，SV 和 SAM 处于分离的状态，SV 的变化不能直接作用到 SAM 上。(2)当待协商的 SA 数量较大时，基于线性链表的 SAM 保存和查找将消耗大量的时间(平均查询时间为 O(n))。该方式 IKE 实现的最大好处是使得 IKE 中 SA(包括 SA 素材)的管理得到保护套件的统一维护；同时，通过 HASH 表的组织，提供了快速查询的基础。该实现结构简单，易于硬件实现。一旦散列摘要部分的哈希计算采用硬件来实现，将显著缩短第三条消息的时间间隔，从而进一步提高 IKE 系统的整体性能。该存储机制的一个示意图如图3所示。

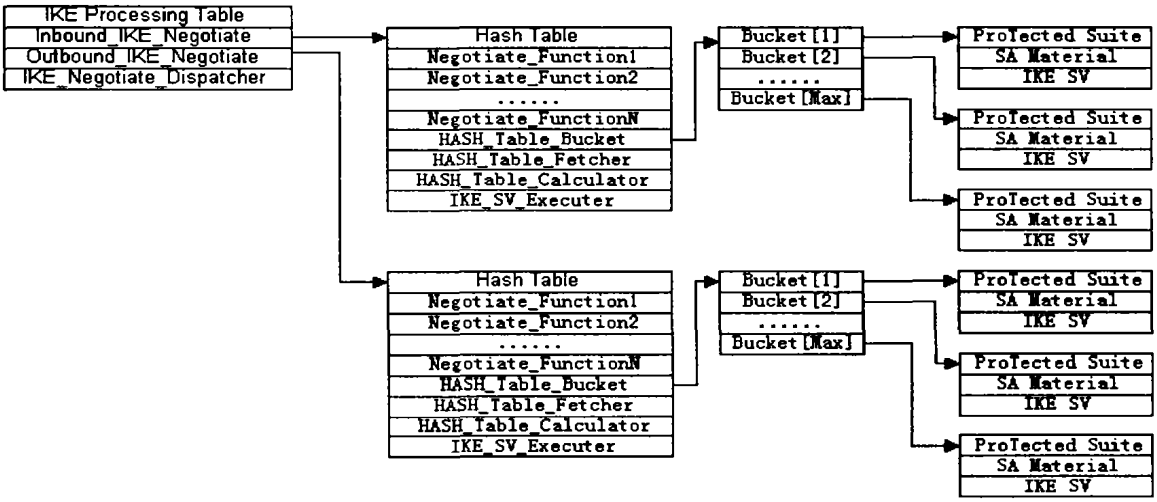

图3 基于 HASH 表存储的保护套件封装机制

从以上分析可以看出，整个 IPSec 安全引擎模型结构灵活，扩展能力强，是一个适于工程实现的高效 HSBR 模型。

总结和展望 HSBR 实现是一个在理论和工程上都有较大意义的课题。随着网络技术的进一步发展，大量原属于 MPP 的技术正在向以分布 NP 为主体的路由器平台迁移。本文分析了 HSBR 的功能和在 HSBR 上实现 IPSec 协议的特点，提出的基于分布并行结构的 HSBR 模型结构灵活、处理快速。可以预期，随着新一代网络处理器和现有微处理器之间的不断融合，以及大量专用芯片的出现，HSBR 将向更高速、更大规模的方向发展，可以提供更快的报文处理速度和更丰富的功能。

参 考 文 献

1 Kent S, Atkinson R. Security Architecture for the Internet Protocol. RFC2401. 1998. 11

2 Kent S, Atkinson R. IP Authentication Header. RFC2402. 1998: 15

3 Kent S, Atkinson R. IP Encapsulating Security Payload (ESP). RFC2406. 1998. 27

4 Harkins D, Carrel D. The Internet Key Exchange (IKE). RFC2409. 1998. 8～12

5 荣霓，龚正虎. 高速边缘路由器中 IPSec 安全引擎实现技术. 计算机工程与科学，2002

6 苏金树. 超高性能路由器. 中兴通讯技术，2001，8：34

7 NetOctave 公司. About the NetOctave FlowThrough Security Architecture. NetOctave NSP4200 Security Processor. 2002

8 汪波. 多处理器系统中高效 Cache 协议的实现方案与模拟：[博士论文]. 2001. 22～27

9 荣霓，龚正虎. 基于强安全通信策略 HLA 改进模型的研究与实现. 计算机研究与发展，2002

(上接第119页)

换。与采用单进程转换模型相比，有以下益处：1)进程 C_P 与 C_Q 可以单独执行，直至需要同步操作时，所以转换器可以并行运行，提高转换效率；2)转换器由两个进程 C_P 与 C_Q 构成，它的状态及转移被分成两部分，所以在调试时，比对单个转换器更容易调试；3)构造的转换器由两个转换进程组成，此方法不仅适合单网关结构，也适合半网关结构。

参 考 文 献

1 Saleh K, Jaragh M. Synthesis of protocol converters: annotated bibliography. Computer Standards &Interface, 1998, 19: 105～117

2 Lam S S. Protocol conversion. IEEE Transactions on Software Engineering, 1988, 14(3): 353～362

3 Takei K, Okumura K, Araki K. Design of gateway system between different signaling protocols of the multimedia session on the Internet. In: The 15th Intl. Conf. on Information Networking, 2001. 297～302

4 Shu J C, Liu M T. A synchronization model for protocol conversion. The 8th Annual Joint Conference of the IEEE Computer and Communications Societies, 1989, 1: 276～284

5 Shu J C, Liu M T. Protocol conversion between complex protocols. In: The 9th Annual Intl. Phoenix Conf. on Computers and Communications, 1990. 584～590

6 Shu J C, Liu M T. An approach to indirect protocol conversion. Computer Networks and ISDN Systems, 1995, 21(2): 93～108

7 Saha D. Verifying the progress properties of a heterogeneous protocol system in an internetworking environment. Computer Communications, 1995, 18(5): 384

8 Dutta S K, Saha D, Das P K. Design of a parallel protocol verification system. The IEEE Region 10 Annual Conference on Speech and Image Technologies for Computing and Telecommunications, 1997, 2: 425～428

9 Hallal H, Negulescu R, Petrenko A. Design of divergence-free protocol converters using supervisory control techniques. The 7th IEEE International Conference on Electronics, Circuits and Systems, 2000, 2(1): 705～708